CSS Services

Naveen Hota

nhota@eos.hitc.com

ECS Release A SDPS/CSMS Critical Design Review
15 August 1995

Roadmap

Communications Subsystem (CSS) Introduction

- Context
- Design Drivers & Approach

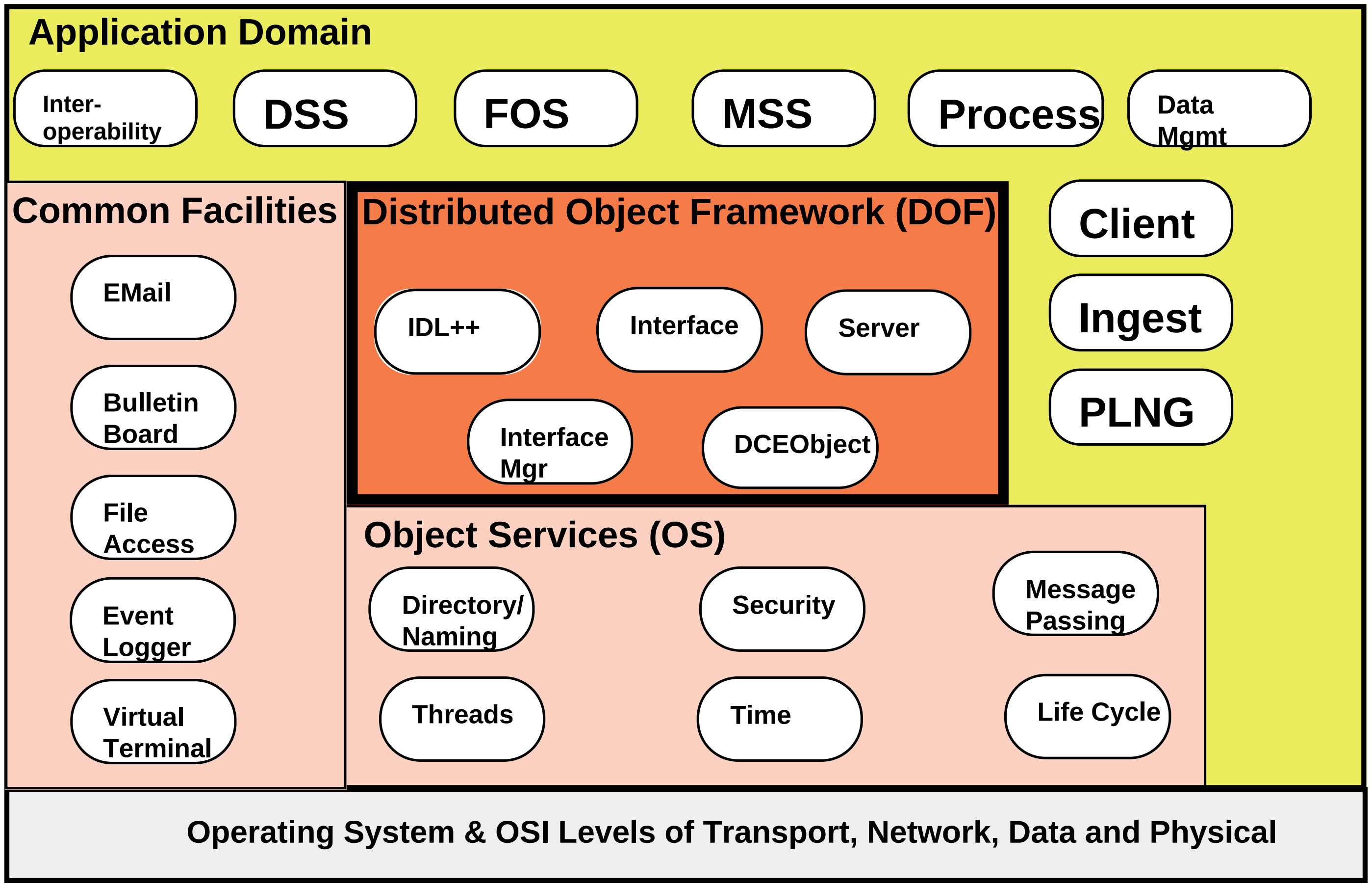
CSS Technology Overview

- Decisions Since PDR
- Technology
- Trades and Prototypes
- Migration

CSS Services

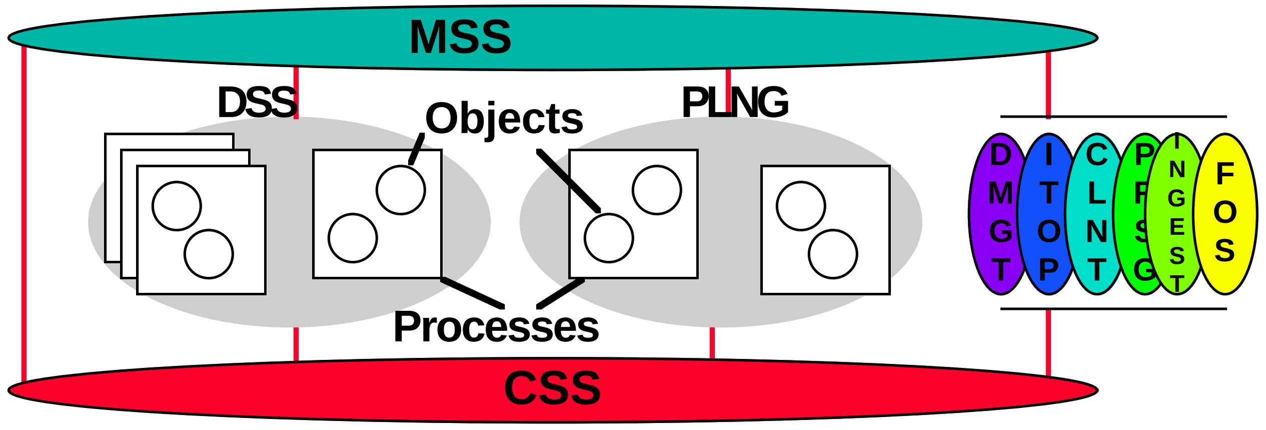
- Distributed Object Framework
- Object Services
- Common Facilities
- Hardware
- Issues
- Wrap-up

CSS Services



Distributed Object Framework

- ECS Application is too large for single address space
- Answer is to break the application into objects and distribute them
- Object Oriented approach
- Facilitates an integrated system view, in spite of the physical distribution
- Provide mechanisms for objects in different processes to communicate
- Distribution is transparent to the end user
- Users: Application programmer



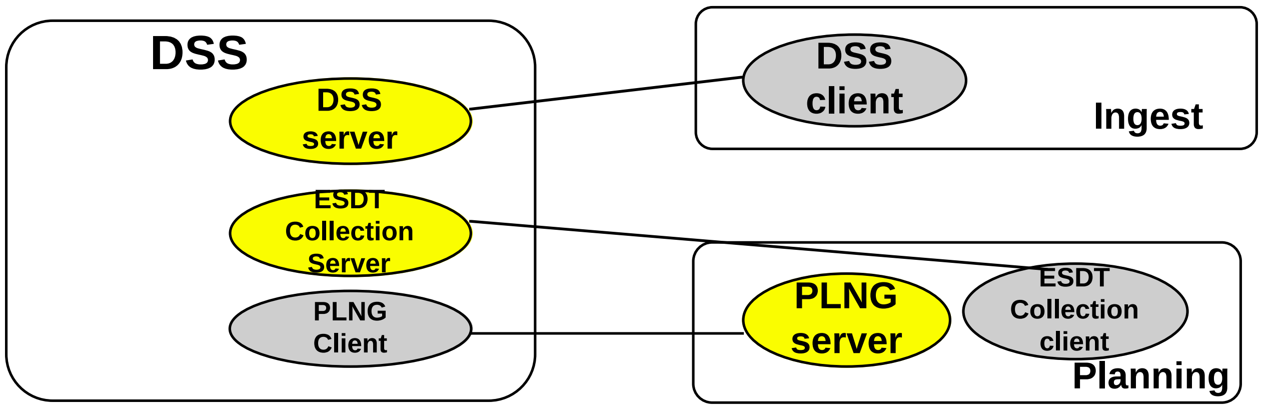
Benefits of DOF

- **Low level communications mechanisms are transparent to the Developer**
- **Separates Interfaces from Implementations**
- **Provides location independence through Directory Service**
- **Provides network based security**
- **Supports OO paradigm**
- **Generic class libraries with default behavior**
- **Customizable by developer for specialized behavior**
- **Transparent interaction with underlying object services**
- **Heterogeneous (Vendor and Platform)**

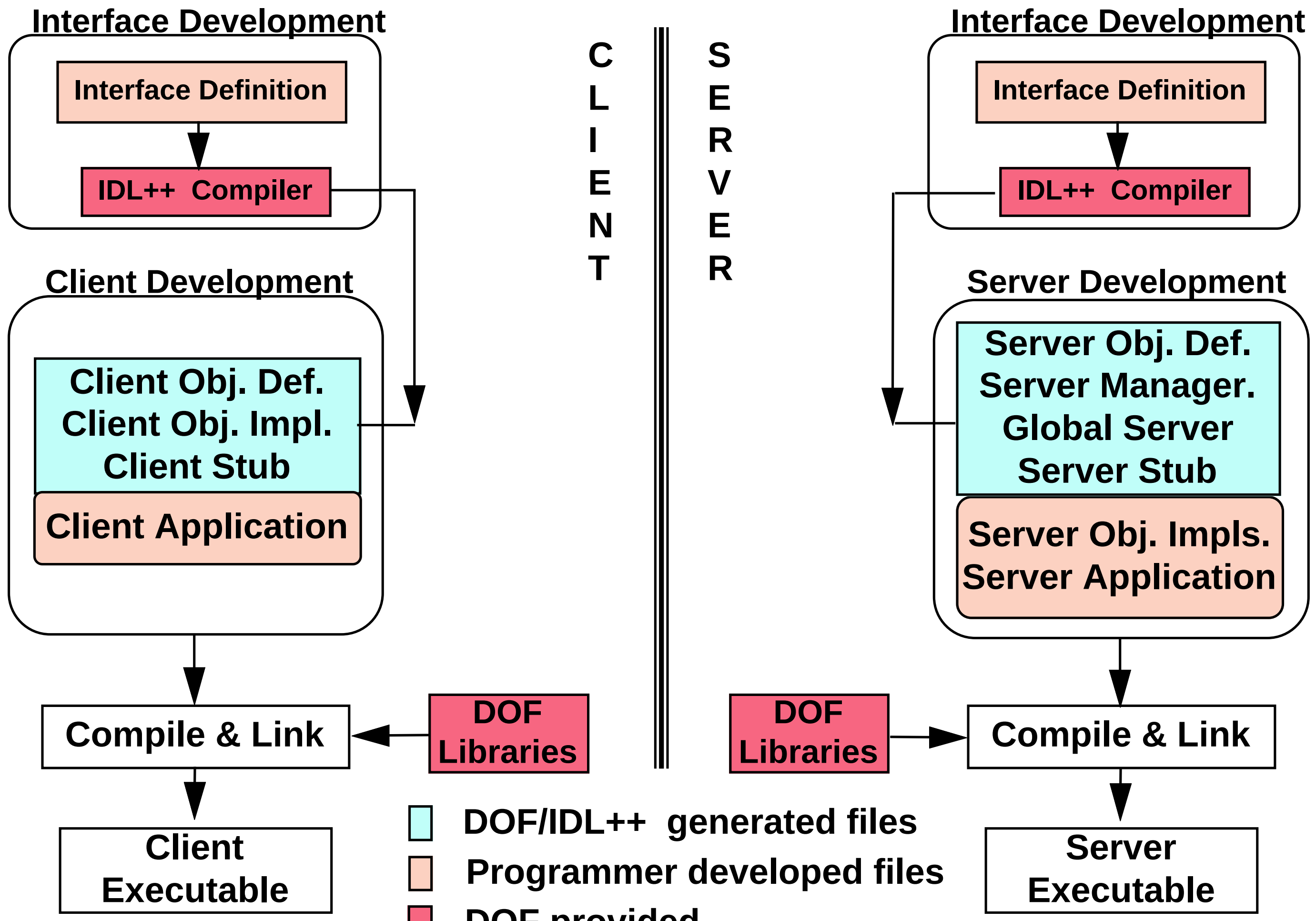
Distributed Application Development

- DOF provides the development environment
- DOF is provided at Release A and utilizes a COTS solution (OODCE)
- A distributed object consists of a client object and a server object
- Object interfaces are written in Interface Definition Language
- IDL++ compiler generates object declarations and communication stubs
- Application programmer implements the class definitions
- Programs are compiled and linked with the DOF libraries and stubs

Example:

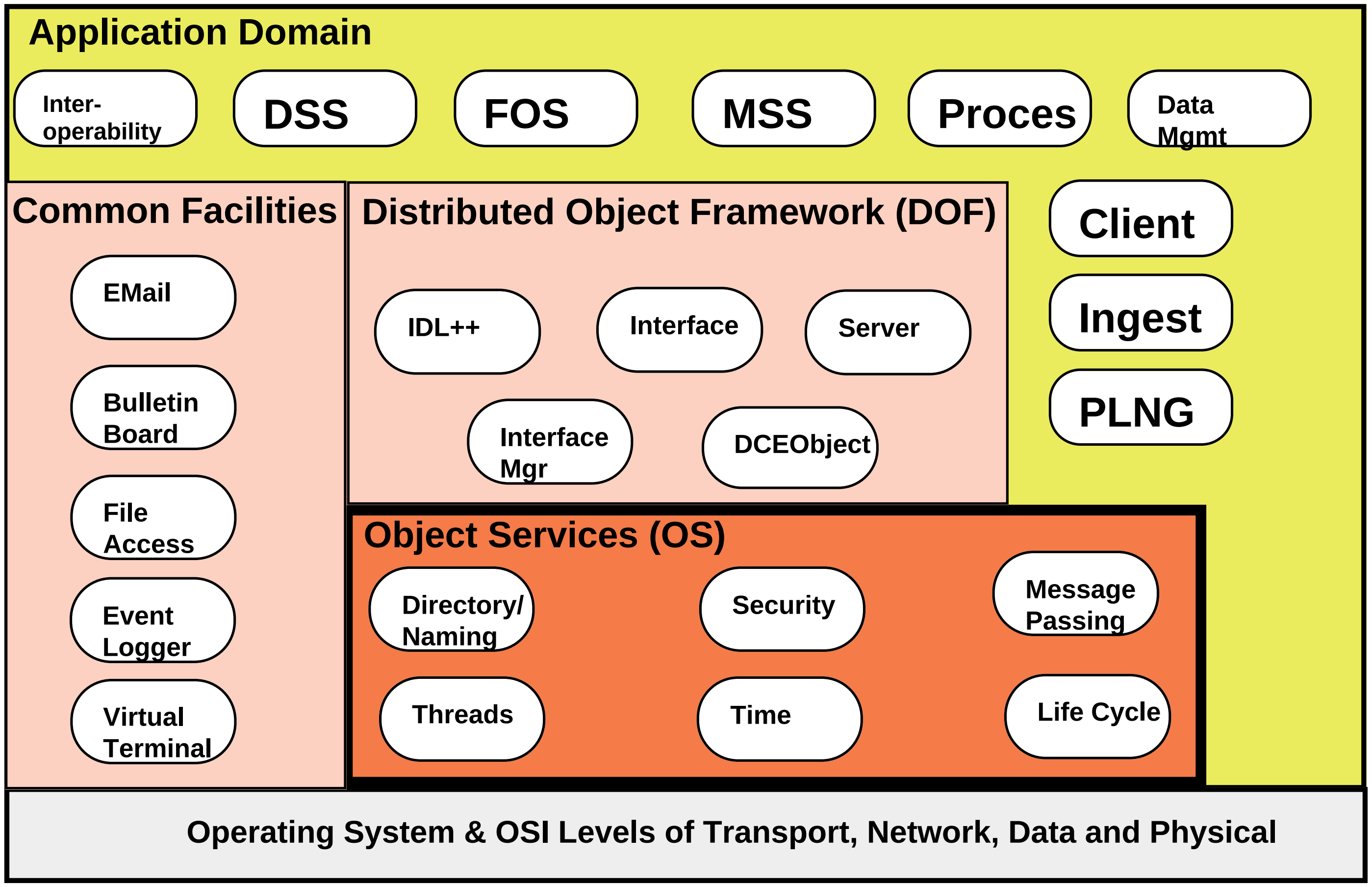


Writing Client/Server Applications



- 🟦 DOF/IDL++ generated files
- 🟧 Programmer developed files
- 🟥 DOF provided

CSS Services



Directory/Naming

Why

- Dynamically locate logical network resources

Functionality

- Provides location transparency
- Allows server applications to store binding information so client applications can find servers
- Stores and retrieves application related information in distributed environment for other applications to share
- Directory information is replicated and distributed across DAACs

Users

- Internal, application programmers

ECS Context

- Stores ECS server (Data Server) binding information
- User account creation (User profiles)
- Asynchronous message passing (logical queue names)
- Multicasting (group names)

Directory/Naming (Cont)

Public Methods - 24

- **DID 305 Vol 12**

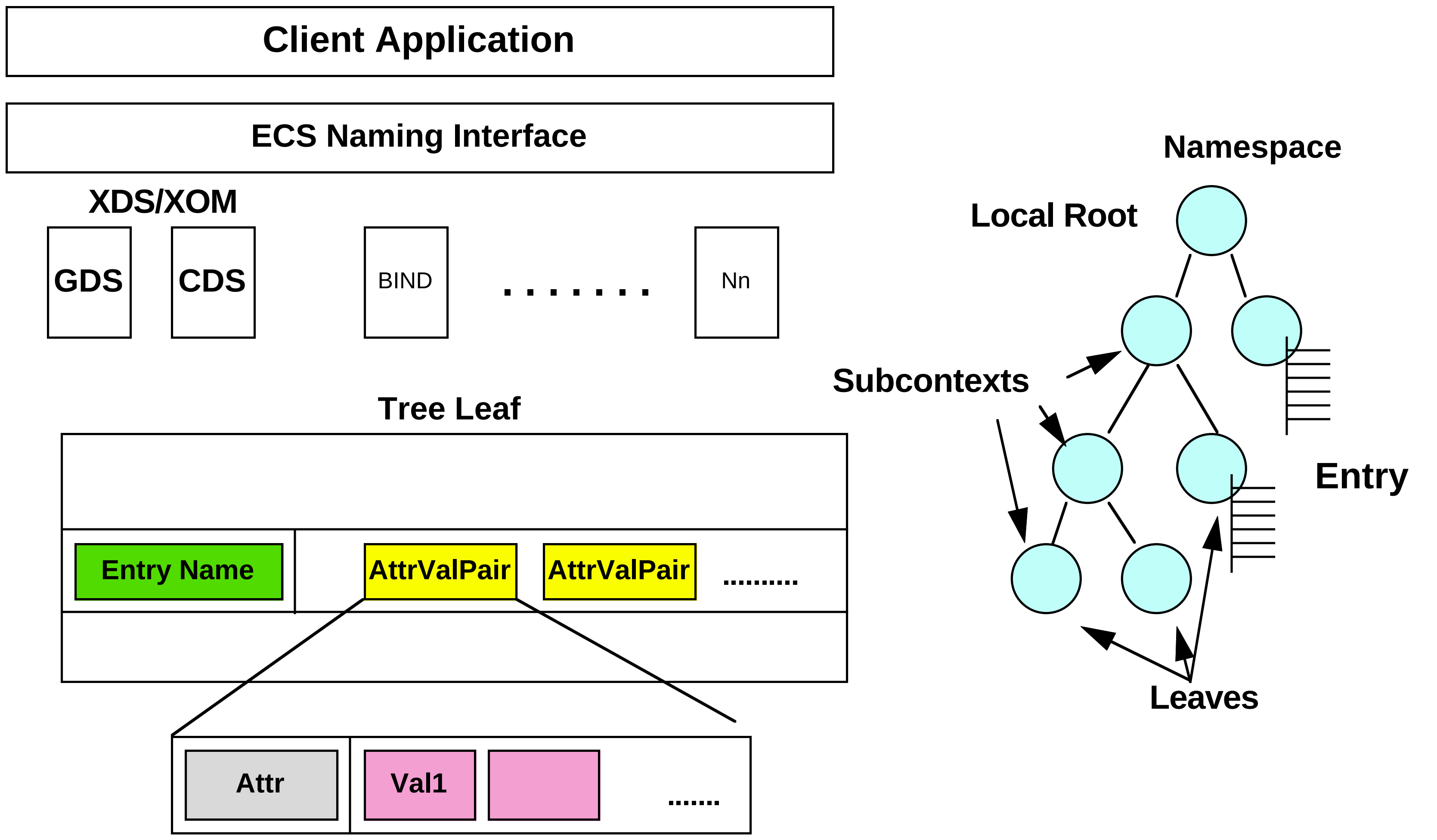
How

- **Uses OSF Cell Directory (CDS) and Global Directory (GDS) Services**
- **An Object Oriented layer on top of the X/Open's XDS/XOM interface to store and retrieve information in DCE Cell and Global Directory Services**

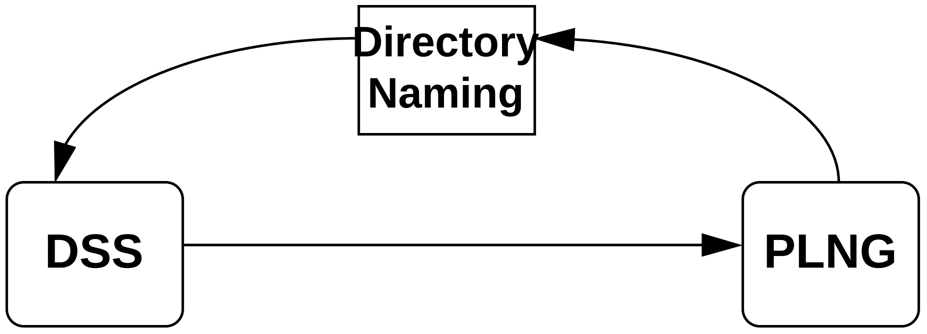
Example

- **Ingest finding the Data Server location and binding information**
- **Data Server finding the location of a logical messaging queue in Planning for subscription/notification purposes**

CSS Directory/Naming Design



Directory/Naming Scenario



- **Planning (and Data Server) creates a queue to receive asynchronous messages and registers it with the namespace**

NOTE: This is the address of the queue within planning and not the planning process itself

- **Planning makes a request to Data Server**
- **Data Server receives it at a later time, processes the request and gets the address of the caller (Planning) from the namespace and returns the results at a latter time**

Time

Why

- **To maintain uniform time across ECS**

Functionality

- **Takes external time and synchronizes host clocks**
- **Simulates time with a specified delta**

Users

- **Internal, application programmers**

ECS Context

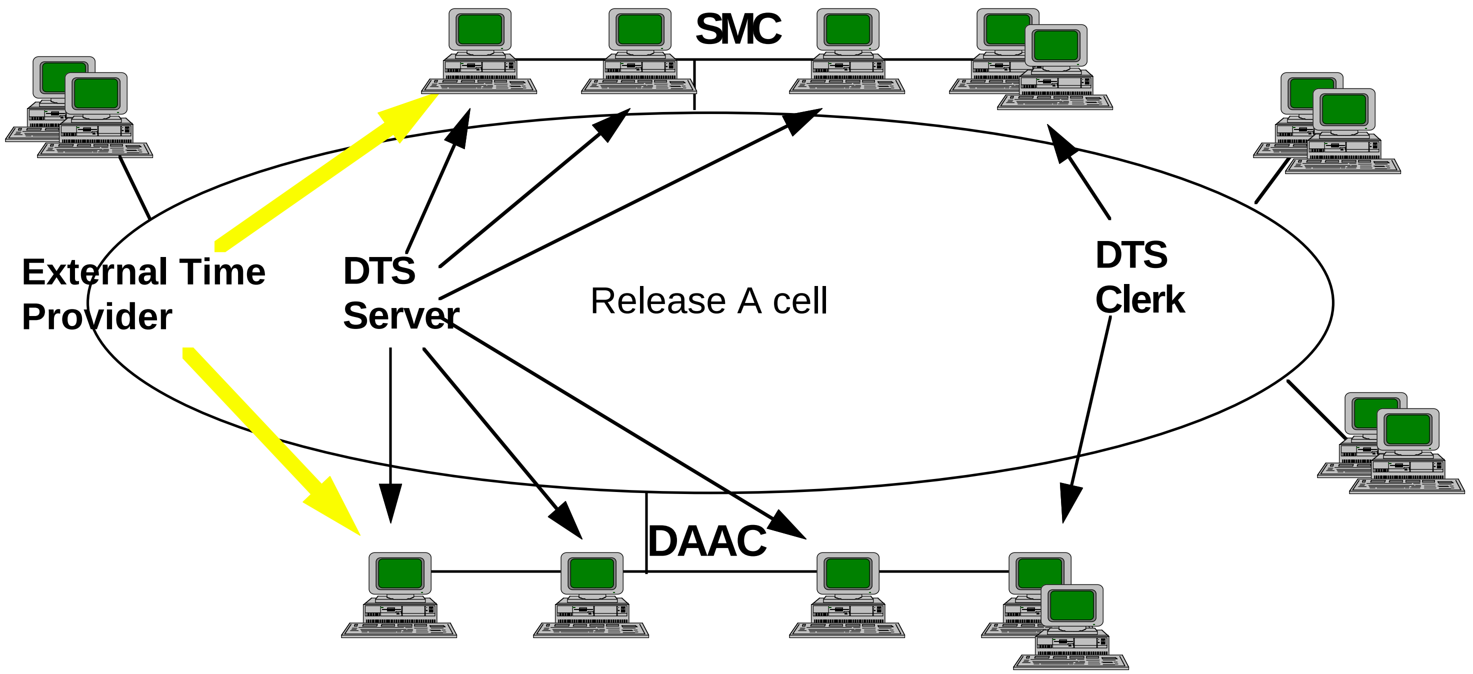
- **Host clocks must be synchronized with reasonable accuracy for event sequencing, duration and scheduling**
- **Distributed event logging**

Public Methods - 14

How

- **Internal synchronization is done via Distributed Time Service (DTS)**
- **Uses (external) time provider at each DAAC**
- **Simulated time is provided by applying delta**

DTS Configuration Plan



- External time (NTP) is fed at each DAAC into one Time Server (Global Server)
- Each LAN will have 3 Time Servers (Couriers)
- Each DCE Client workstation will have a time clerk

Message Passing

Why

- **To provide asynchronous communications between ECS services**

Functionality

- **Control returns to the caller immediately**
- **Provides store and forward mechanism**
- **Guaranteed message delivery with callbacks**
- **Provides multiple priority levels**
- **Multiple number of retries with intervals (specified)**

Users

- **Application programmers**

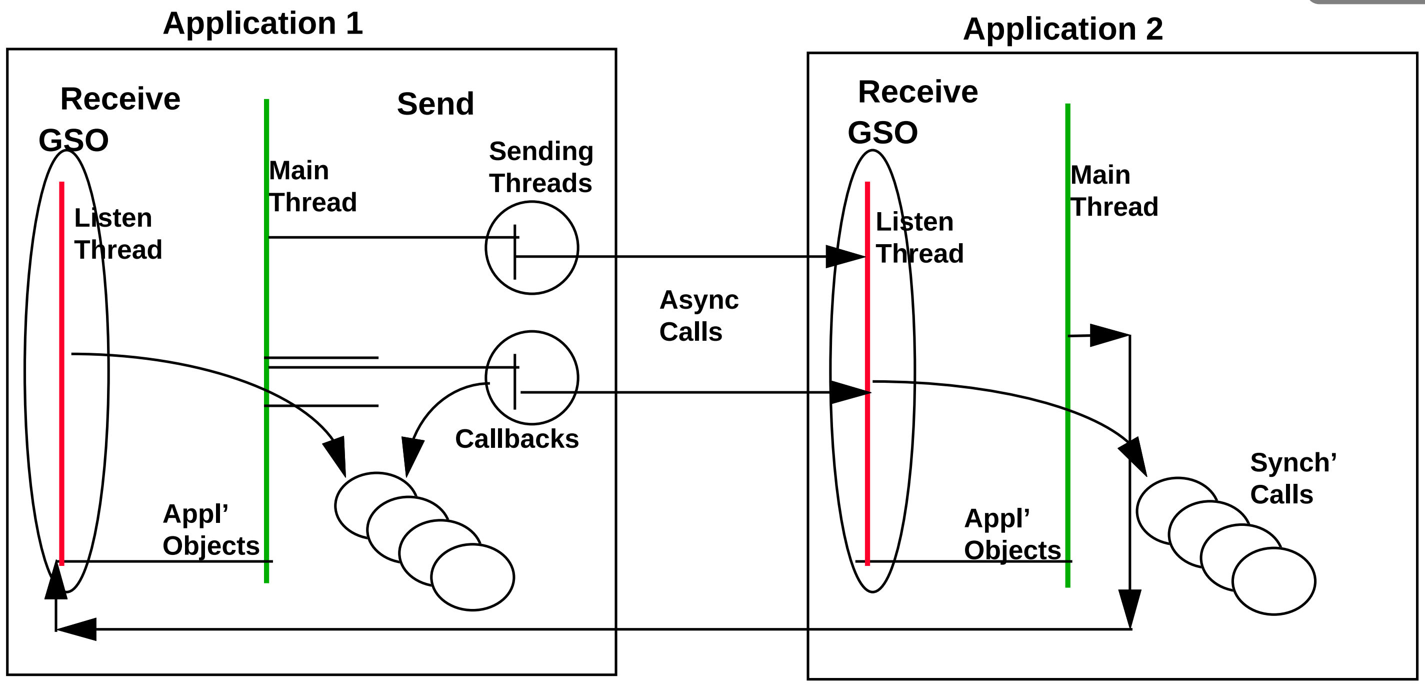
ECS Context

- **Subscriptions (Data Server, Planning, FOS data)**
- **Notifications (Management Applications)**

Two Methods

- **Simple Message Passaging & Persistent Message Passing**

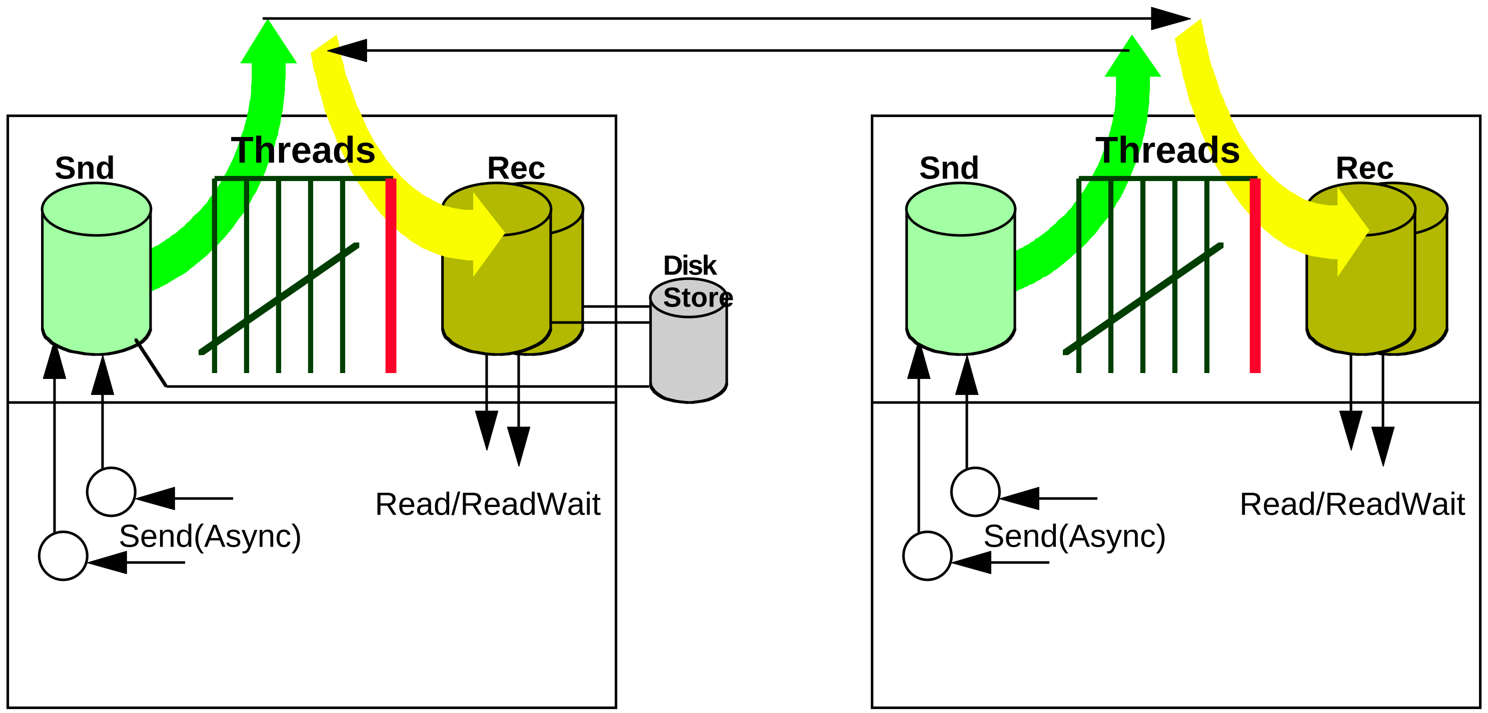
Simple Message Passing



- Public Methods - 26
- No changes on server side
- Remote method invocation with multiple argument types
- No store and forward

Example: Agent Notifications to the Network Node Manager

Persistent Message Passing



- **Public Methods - 52**
- **Transfers byte streams**
- **Store and forward with persistence**

Example: DSS Notifications to Planning that a certain type of data granule is inserted

Security

Why

- To protect the integrity of ECS data and services (resources)

Functionality

- Creates, maintains and verifies user/server identities
  - Server keytab files (passive principals)
- Creates, maintains and checks privileges for service access
  - Create and maintain Access Control Lists (ACL)
  - Provides persistence
- Protects data in transit

Users

- Internal, application programmers

ECS Context

- Authenticates ECS users and Servers
- Authorizes user/client access to services/resources

Public Methods - 53

Security (Cont)

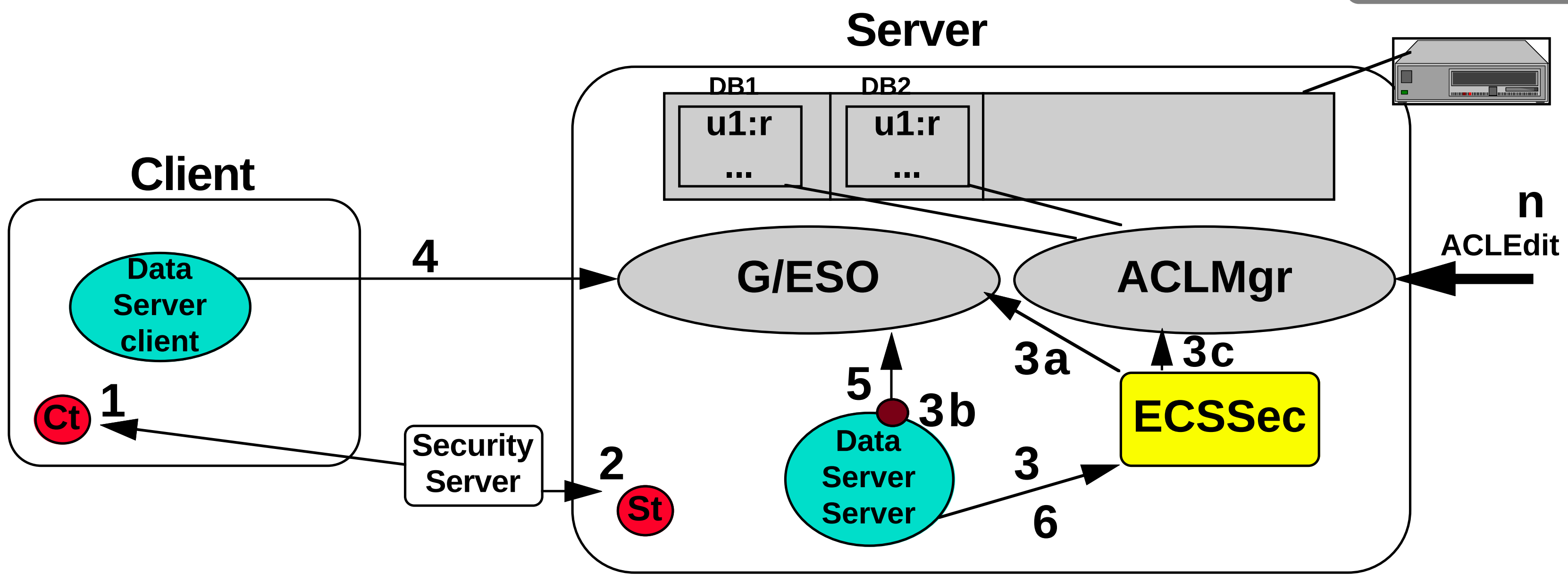
How

- **A layer of encapsulation on top of OODCE provided classes**
- **Specialization of OODCE classes for server identities and persistence of ACLs**

Example:

- **Security initialization for a Data Server**
- **Authorization process at a Data Server**

Client/Server Security Interaction



1: Client gets security tickets from the Security Server
2: Data Server sets server identity (Keytab files) and gets tickets
3: Data Server sets preferences (data, access privileges)
4: User (Client) checks client preferences
5: Data Server checks server preferences
6: Data Server method (CreateESDTCollection) checks client privileges
n: M&O edits user privileges through the external interface (acledit)

Lifecycle

Why

- To control ECS resources remotely

Functionality

- Provides application control
- Creating and deleting distributed objects on demand within applications

Users

- Application Programmers

ECS Context

- All ECS applications for startup / shutdown / suspend / resume
- Some applications to create new distributed objects on demand
  - Data Server ESDT Collection objects for individual users
  - Data Server Configuration object

Public Methods: 8

Lifecycle (Cont)

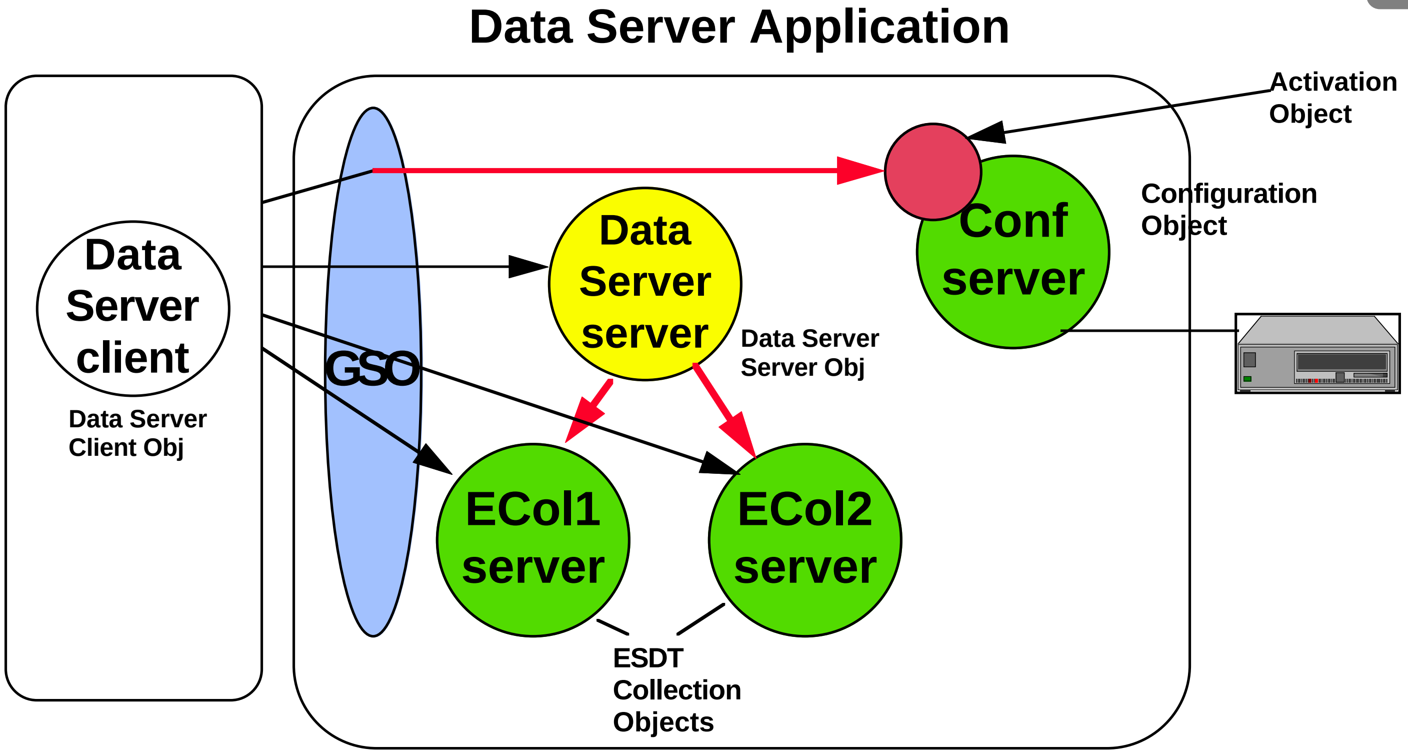
How

- **An Activation Object monitoring the state of an object (provided by OODCE)**
- **Multiple instances of an object can be created through Factories (Programmer)**
- **Specialize the Global Server Object for control services**
  - **Graceful Shutdown, Suspend, Resume of a server application**

Example:

- **Data Server Configuration object gets activated and deactivated on demand**
- **Data Server creating Distributed ESDT Collection objects through Factories for each user session**

Factories & Activation



- Activation activates the same object (brings the object into memory from disk)
- Factories create new objects

Threads

Why

- **To improve performance**

Functionality

- **Provides parallelism in processing**

Users

- **Internal, application programmers**

ECS Context

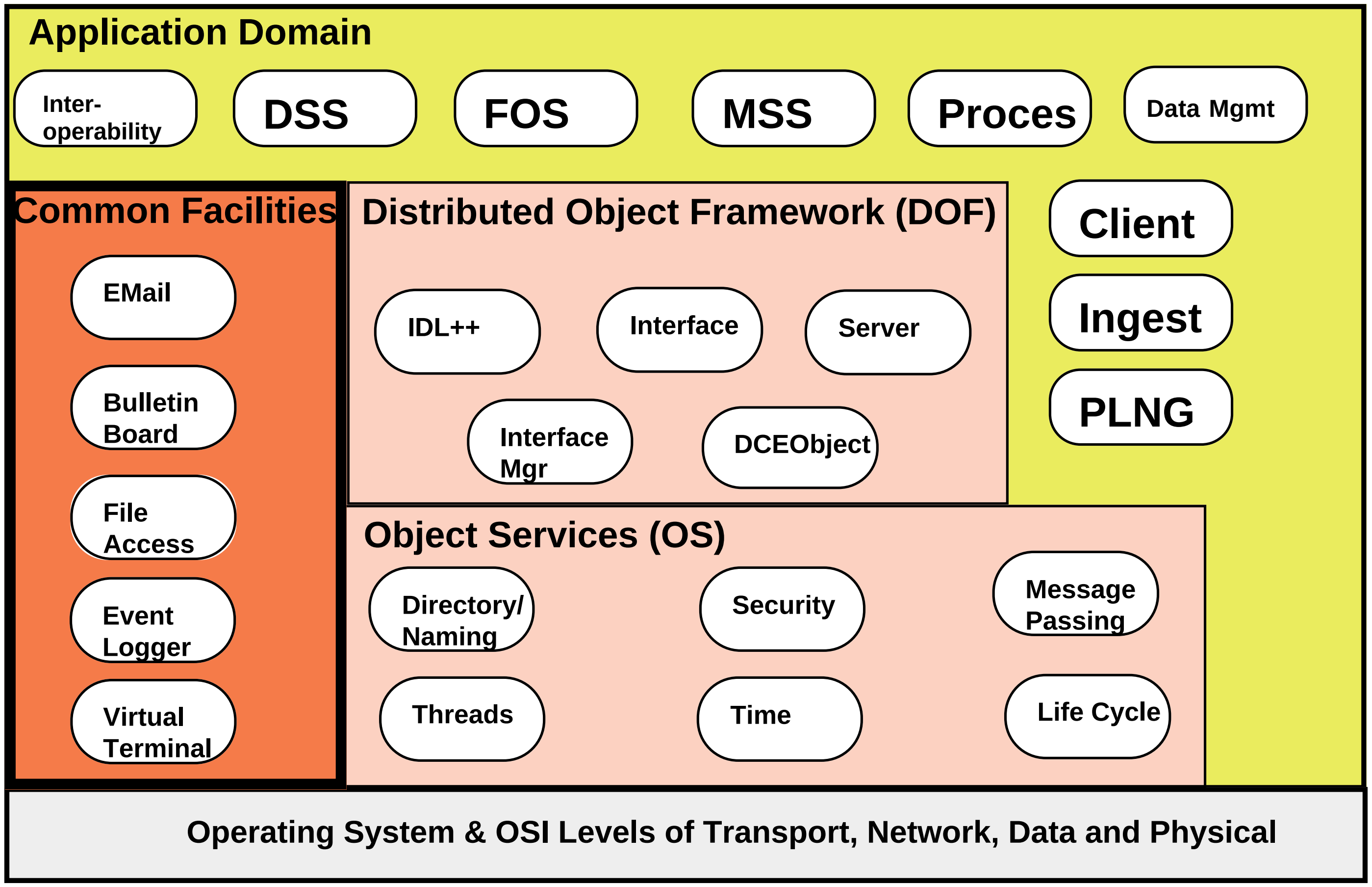
- **All server applications need threads (Internal)**
- **Applications need this for concurrent processing**

Public Methods - 41

How

- **DCE/OODCE Threads**

CSS Services



Electronic Mail

Why
- Operators/applications to communicate with users

Functionality
- Operators will have software to interactively read and send messages
- The application developers will have an API which they can use to send messages

Users
- Operators, application programmers, end users

ECS Context
- Operators use E-Mail to interact with ECS users
- End users use it to interact with ECS only
- Data Server sends mail messages to users upon completion of acquire requests

How
- COTS software (ZMail) will be used for the operators
- CSS provides API for the application developers to send E-Mail

Public Methods - 22 (Including Bulletin Boards)

Bulletin Board

Why

- To share ECS information electronically among distributed users

Functionality

- A common place to share (post and read) information messages
- API to post messages to the Bulletin Board(s)

Users

- End users

ECS Context

- ECS users for ECS related information

Public Methods - 14 (including E-Mail)

How

- The implementation will be COTS (NNTP server)
- CSS will develop API needed to post messages to bulletin boards
- Client access through WEB browser

FTP

Why

- To transfer data electronically within ECS and to external entities

Functionality

- Transfers files interactively
- Transfers files programmatically (API)
- Provides authenticated access [via keberized FTP (kFTP)]
- Provides Notification capability when a file transfer is complete

Users

- Ingest, Data Server, SCF, data providers, applications

ECS Context

- Ingest uses k/FTP pull to get files from external data providers
- SCFs would transfer source (algorithms) through interactive FTP
- Data Server uses FTP notification to know that user has retrieved a file

Public methods - 2

FTP (Cont)

How

- **The implementation will be COTS (FTP and kFTP)**
- **CSS will develop API needed to transfer files between the application and the COTS FTP client**
- **Modify FTP Server (at ECS) to notify ECS applications when a file is retrieved**

Event Logging

Why

- To generate a permanent log of ECS events

Functionality

- CSS provides a set of objects to allow developers to log messages
  - to local files
  - to management logs with criteria to trigger SNMP traps

Users

- Application programmers

ECS Context

- ECS applications to log events for historical data

Implementation

- Class library implementation with MSS interfaces to SNMP trap
- Custom

Public Methods - 22

Virtual Terminal

Why

- **To allow remote login sessions into designated ECS hosts**

Users

- **SCFs, operators, data providers**

ECS context

- **SCFs to remotely log into designated ECS hosts to correct algorithms**

How

- **Telnet/kTelnet**

Universal Reference

Why

- Need to <u>save</u> and locate ECS resource information

Functionality

- provide persistent identifiers

Users

- Data Server, Advertiser, Planning, Processing

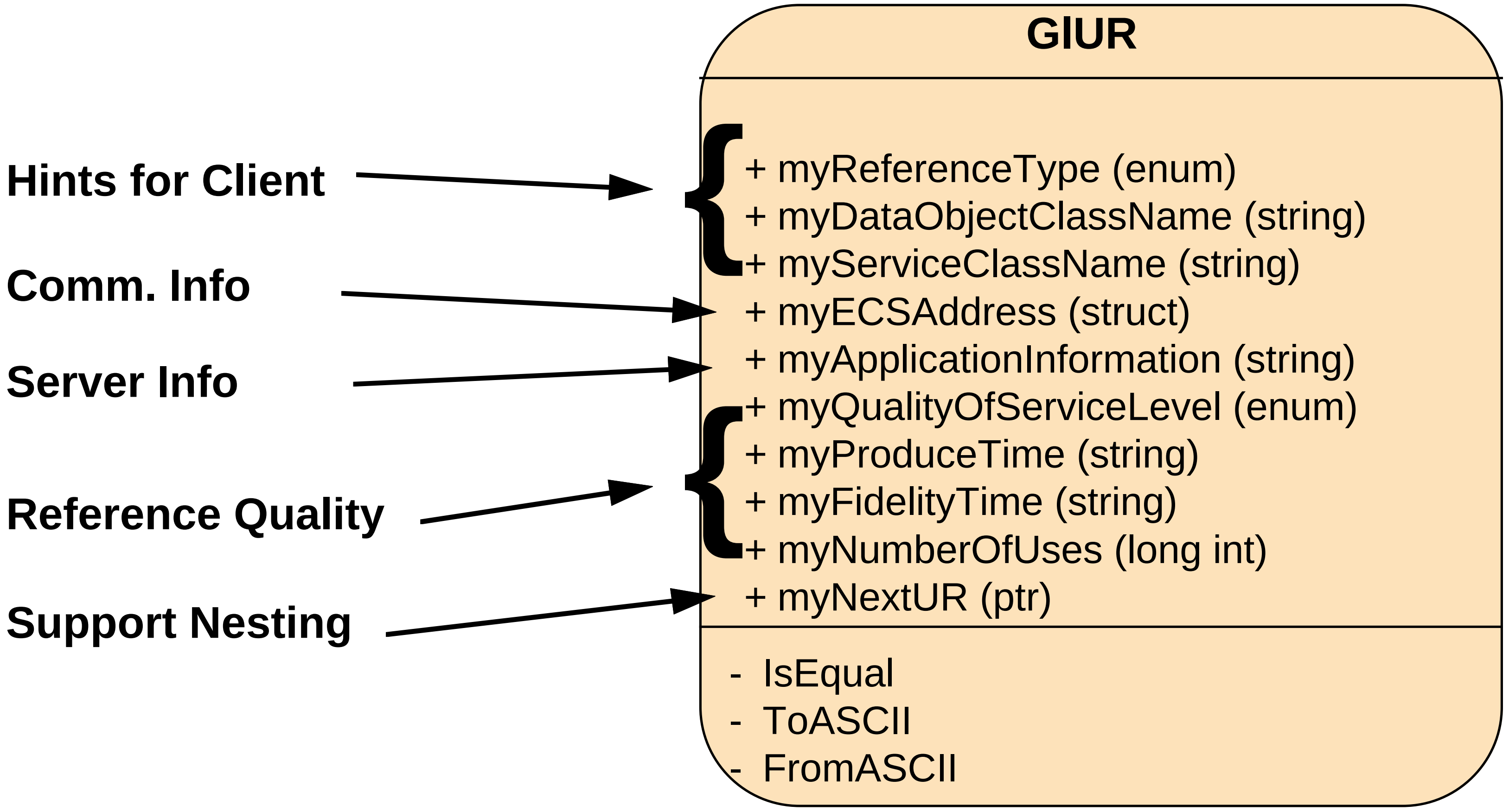
ECS Context

- Data Server creates URs for data granules
- Advertiser maps URs to ECS services
- Planning uses for data availability and data checking
- Processing uses to stage data

How

- Class library implementation

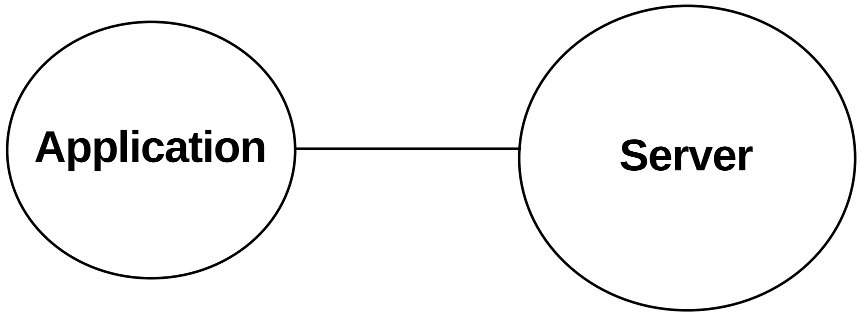
UR Design


Hints for Client
Comm. Info
Server Info
Reference Quality
Support Nesting
GlUR
+ myReferenceType (enum)
+ myDataObjectClassName (string)
+ myServiceClassName (string)
+ myECSAddress (struct)
+ myApplicationInformation (string)
+ myQualityOfServiceLevel (enum)
+ myProduceTime (string)
+ myFidelityTime (string)
+ myNumberOfUses (long int)
+ myNextUR (ptr)
- IsEqual
- ToASCII
- FromASCII

UR Scenario

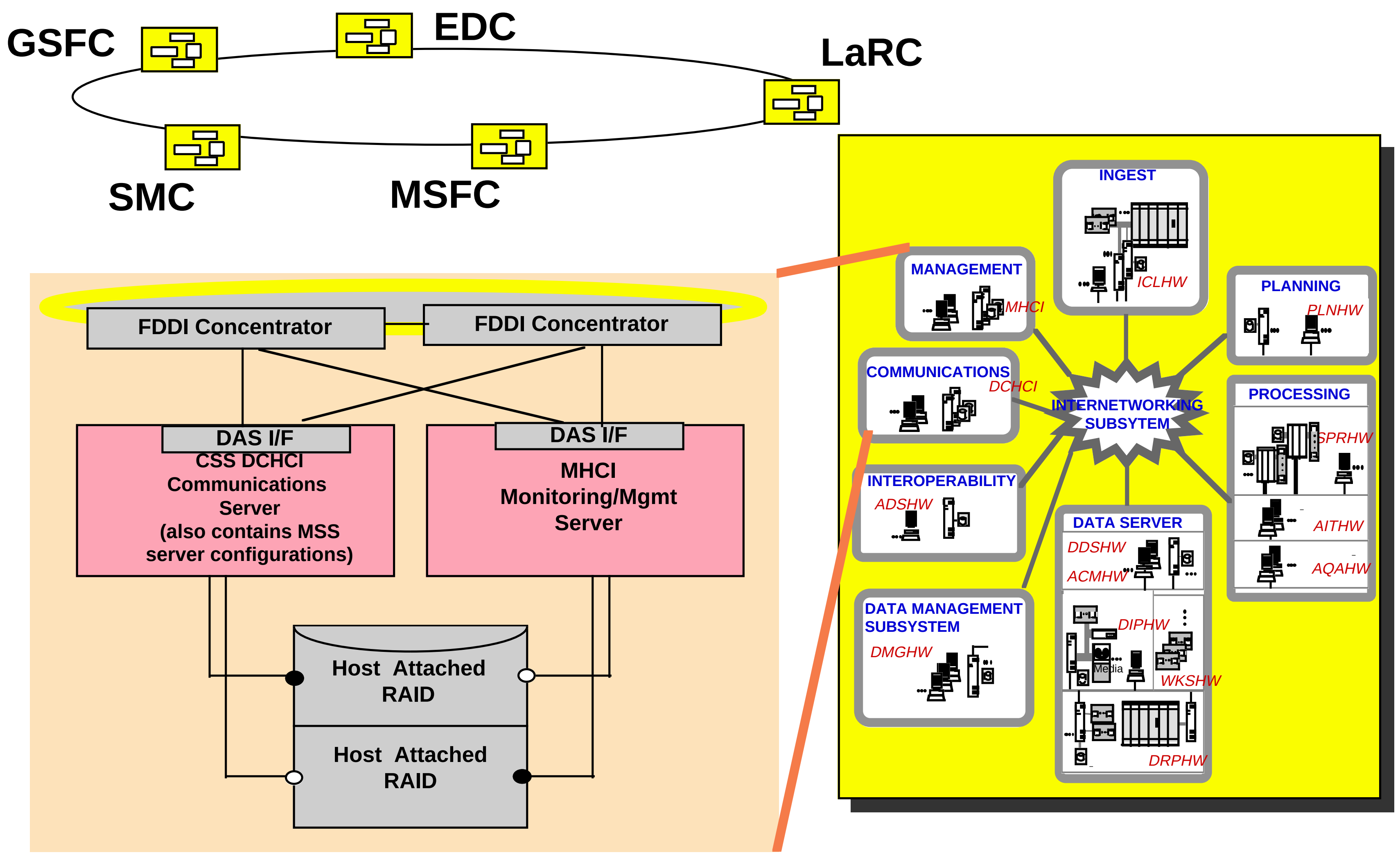
Server:
Obtains Comm. Handles at Start-up Time

Appplication:
Issues Search
Server:
Finds Objects (e.g., Data Granules)
Creates UR (With Comm. Handles)
Packages Granule Identifier in UR
Returns Object Attributes, Including UR
Application:
Using Server "Proxy", Obtains/Saves UR



Appplication:
Gets Saved UR
Uses "Service Type" to Determine What API to Call
Creates Server "Proxy"
Proxy:
Determines Correct Server
Application:
Issues Retrieval Request
Server:
Unpacks Granule ID

CSS Hardware Architecture



CSS Services Implementation

Services	COTS (W/Glue)[1]	Custom
Object Services		
Directory / Naming	BIND - DNS; CDS/GDS - DCE	Extensions to CDS/GDS on top of XDS/XOM
Security	OODCE class libraries - Authentication, Authorization, Integrity, Privacy and Keytab files	Server Identities a layer on top of OODCE for authorization and persistence of ACLs
Message Passing	none	Message passing on top of OODCE with guaranteed delivery, callbacks, priorities, store and forward features
Thread	DCE/OODCE	none
Time	DCE/OODCE	Time skew (delta)
Lifecycle	OODCE	Control mechanisms for MSS
Distributed Object Framework (DOF)		
DOF Services	OODCE	Changes in global server object
Common Facilities		
E-Mail	COTS (ZMail)	API to send messages
Bulletin Board	COTS (NNTP server)	API to post messages
File Access (ftp kftp)	COTS	API to send / receive messages Modify ftp server to notify ECS applications upon transfer of files
Virtual Terminal	COTS	
Event Logger		Class libraries

[1] COTS (W/Glue) - requires some "glue" code

Issues

Issue: CSS Performance Overhead was identified as Risk Item at PDR

Concern: Encapsulation of OODCE (baselined at PDR) could cause excessive overhead

Strategy: Performance evaluations planned during Ir1. This will allow time to tweak OODCE performance before Rel. A is operational

Avoid encapsulation unless absolutely necessary

ECS Actions: Perform Benchmark testing

Revised analysis showed encapsulation of OODCE should be dropped.

Monitor Ir1 performance

ECS Benefit: Controlled performance

Issues

Issue:	OODCE Availability by Platform was identified as Risk Item at PDR
Concern:	Proprietary OODCE not available on all platforms
Strategy:	Ensure OODCE ports are available for required platforms
ECS Actions:	HP and SUN (Solaris) ports are operational Agreements in place for SGI and DEC ports - ready mid Nov 95 and Feb 96 respectively
ECS Benefits:	OODCE available for deployment as needed

Issues

Issue: Need Migration to 'Object Request Broker' services in later releases.

Concern: Method should not be too costly or cause excessive code breakage in Release A and B

Strategy: Develop alternative migration paths

Have a clear demonstratable migration path that will incur minimal code breakage cost

ECS Actions: Prototype CORBA 2.0 products to verify migration and assess costs.

Prototype DCE 1.2 (when available) to verify migration

ECS Benefit: Reduced migration costs/breakage

CSS Wrap-Up

- CSS Context, Design Approach, Decisions since PDR
- Cell topology/deployment
- Services as required by ECS
- Mostly COTS implementation
- Standards based
- Evolvable / Migratable
- Incremental development & prototyping
- Technology

Glossary

ACL	Access Control List
AFS	Andrew File System
ATM	Asynchronous Transfer Mode
API	Application Programming Interface
BB	Bulletin Board
BBS	Bulletin Board Service
BIND	Berkeley Internet Name Domain
CDS	Cell Directory Service (part of DCE)
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off The Shelf
CSMS	Communication and Systems Management Segment
CSS	Communication SubSystem
DAAC	Distributed Active Archive Center
DCE	Distributed Computing Environment (from OSF)
DFS	Distributed File System (part of OSF/DCE)
DNS	Domain Name System
DOF	Distributed Object Framework (CSS infrastructure)
DTS	Distributed Time Server (part of DCE)
ECS	EOSDIS Core System
EMail	Electronic Mail
EP	Evaluation Prototype
FOS	Flight Operations Segment
ftp	File Transfer Protocol
GDS	Global Directory Service
http	HyperText Transfer Protocol
IDL	Interface Definition Language
IIP	Internet Protocol
ISS	Internetworking SubSystem
kerberos	Security protocol developed by MIT; base for DCE security
kftp	Kerberized File Transfer Protocol
ktelnet	Kerberized telnet
M&O	Maintenance and Operations
MR-AFS	Multi -Resident Andrew File System
MSS	Management SubSystem
NFS	Network File System
NNTP	Network News Transfer Protocol
NTP	Network Time Protocol
OMT	Object Modeling Techniques
OMG	Object Management Group
OO	Object Oriented
OODCE	Object Oriented DCE - HP product
ORB	Object Request Broker
OS	Object Services (CSS building blocks)
OSF	Open Software Foundation
OSI	Open System Interconnect
RFA	Remote File Access
RMP	Reliable Multicast Protocol
RPC	Remote Procedure Call
SCF	Science Computing Facility
SDPS	Science Data Processing Segment
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
TCP	Transport Control Protocol
UDP	User Datagram Protocol
WWW	World Wide Web
X.500	CCITT Standard for Naming